НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки”

на тему: Хмарний сервіс швидкого прототипіювання

Виконала: студентка 4 курсу, групи ТР-52

Бойко Інеса Вячеславівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Демчишин Анатолій Анатолійович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту
Бойко Інесі Вячеславівні
(прізвище, ім’я, по батькові)

1. Тема роботи “Хмарний сервіс швидкого прототипіювання”

керівник роботи доцент, к.т.н. Демчишин Анатолій Анатолійович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”___”_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.

3. Вихідні дані до роботи розрахунок об’єму моделі, маси, довжини нитки для друку ABS пластиком, приблизна вартість та вікно з завантаженою моделлю.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати проблематику прототипування та хмарних сервісів, провести порівняльний аналіз сучасних технологій програмування для реалізації клієнт-серверної архітектури хмарних рішень, створити сервіс для проведення розрахунку витрат матеріалу для друку 3D моделі використовуючи ABS пластик.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов’язкових креслень) 1. Мета та завдання роботи. 2. Огляд існуючих рішень. 3. Математичні методи розв’язку задачі. 4. Функції хмарного сервісу швидкого прототипування. 5. Схема роботи додатку. 6. Інтерфейс користувача. 7. Висновки.

Дата видачі завдання ”___”_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	Передзахист		
9.	Захист		

Студентка _____ Бойко І.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Демчишин А.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою роботи було створення хмарного сервісу швидкого прототипування, що дасть змогу користувачу завантажити свою готову 3D модель у форматі STL та на її основі проводити розрахунок витрат матеріалу при подальшому друку ABS пластиком. Всі обчислення проводяться на стороні сервера повертаючи клієнту отриманий результат. Користувацький додаток представляє собою демонстрацію завантаженої користувачем моделі у окремому вікні, виведення результатів обчислень моделі або у разі помилки файлу відповідне повідомлення.

Ключові слова: прототипування, хмарний сервіс, Node.js, REST, Ajax, STL, 3D друк, ABS пластик.

Записка містить 63 сторінки, 32 рисунки, 5 формул та 20 посилань.

ABSTRACT

The goal of the work was to create a cloud-based rapid prototype service that will allow the user to download their finished 3D model in STL format and, on its basis, to calculate the material costs for further printing with ABS plastic. All calculations are performed on the server side by returning the result to the client. The custom application is a demonstration of a user-loaded model in a separate window, the output of the calculation model results, or in the event of a file error, the corresponding message.

Keywords: prototyping, cloud service, Node.js, REST, Ajax, STL, 3D printing, ABS plastic.

The note contains 63 pages, 32 images, 5 formulas and 20 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – Hyper Text Markup Language, стандартна мова розмітки для створення веб-сторінок і веб-додатків.

CSS – Cascading Style Sheets, формальна мова для описання зовнішнього вигляду документу написаного з використанням мови розмітки.

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.

REST – Representational State Transfer, підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

Ajax – Asynchronous JavaScript And XM, підхід до побудови користувацьких інтерфейсів веб-застосунків.

Canvas – елемент HTML5, який можна застосовувати для малювання графіки використовуючи скрипти (переважно JavaScript).

STL – stereolithography, формат файлу, широко використовується для зберігання тривимірних моделей об'єктів для використання в адитивних технологіях.

VDS / VPS – Virtual Dedicated Server / Virtual Private Server, хостинг-послуга, де користувачеві надається віртуальний сервер з максимальними привілеями.

ВСТУП

В період технічного та інформаційного прогресу майже всі сфери людського життя зводяться до використання різноманітних онлайн-сервісів. Не виключенням є і виробництво, одним з найважливіших етапів якого – є створення прототипів предметів або конструкцій. Саме на цій стадії інженер має можливість оцінити дизайн, ергономічність та доцільність розробки, перед тим, як витрачати ресурси на її виробництво. Для отримання готового макету перед друком створюється його комп'ютерна CAD-модель в STL форматі.

З розвитком технологій хмарних обчислень стала можливою поява і хмарних сервісів, що дало змогу користувачам поєднувати внутрішні ресурси свого комп'ютерного пристрою та програмні ресурси, які надають йому інтернет-сервіси [1]. На відмінну від загальновживаних додатків 3D моделювання (Rhino, Blender), хмарний сервіс дає можливість працювати без процесу інсталяції, оновлення програмного та апаратного забезпечення, що значно зменшуєш поріг входу для початку роботи [2].

Метою роботи є створення хмарного сервісу швидкого прототипування, що дає змогу користувачу проводити розрахунок деяких величин моделі. Дана система на сьогоднішній день є актуальною, адже її використання дозволяє визначити об'єм завантаженої моделі, її масу, а також приблизну вартість при друку ABS пластиком.

У роботі описується повний цикл розробки сервісу, починаючи від вибору та огляду технологій для його написання, налаштування необхідного програмного забезпечення та завершуючи описом використання системи користувачем.

Зміст розділів даної роботи наступний:

У першому розділі описується постановка задачі реалізації системи, її мета, об'єкт і предмет дослідження, а також завдання дослідження.

У другому розділі приведений огляд проблематики прототипування та хмарних сервісів, а також описано реалізацію сервісу швидкого прототипування.

У третьому розділі проводиться огляд технологій програмування для реалізації клієнт-серверної архітектури хмарних сервісі, а також наведені критерії їх вибору та аналіз.

У четвертому розділі описується налаштування середовища Node.js та підключення клієнтських бібліотек для роботи з 3D об’єктами у браузері.

У п’ятому розділі описується програмна реалізація системи прототипування.

У шостому розділі описується сценарій взаємодії користувача з системою та її недоліки.

1. ПОСТАНОВКА ЗАДАЧІ РЕАЛІЗАЦІЇ ХМАРНОГО СЕРВІСУ ШВИДКОГО ПРОТОТИПУВАННЯ

Метою розробки системи є надання можливості розрахунку вартість друку 3D моделі ABS пластиком, а також обчислення її об’єму, маси та довжини нитки необхідної для друку.

Об’єктом дослідження є хмарні технології та 3D моделювання.

Предметом дослідження є сервіс швидкого прототипування.

Для досягнення поставленої задачі були сформульовані наступні завдання дослідження, що визначили логіку дослідження та його структуру:

— проаналізувати проблематику прототипування та хмарних сервісів;

— провести порівняльний аналіз сучасних технологій програмування для реалізації клієнт-серверної архітектури хмарних рішень;

— створити сервіс для проведення розрахунку витрат матеріалу для друкування 3D моделі використовуючи ABS пластик.

Вхідною інформацією вважаються наступні дані:

— завантажена користувачем 3D модель у форматі STL.

Вихідною інформацію вважаються наступні дані:

— відображена у вікні браузера модель користувача;

— розрахунок об’єму моделі;

— розрахунок її маси;

— довжина нитки для друку моделі;

— приблизна вартість друку ABS пластиком.

Сервіс повинен бути зрозумілий та зручний у використанні для користувача будь-якого віку.

Система повинна мати такі функції:

— можливість завантажити файл з 3D моделлю;

— можливість отримати модель у окремому вікні;

— можливість зміни розташування камери аби мати можливість більш краще оглянути модель;

— отримання об’єму завантаженої моделі, її маси, довжини нитки для 3D друку та приблизну вартість при друці ABS пластиком;

— зрозумілі повідомлення про помилки при користуванні сервісом користувачем.

В якості інструменту для створення сервісу було обрано програмну платформу Node.js. Дана платформа дає змогу максимально ефективно втілити REST (Representational State Transfer) сервіс.

Крім того, для створення клієнтського інтерфейсу була використана технологія Ajax (Asynchronous Javascript And Xml), яка дає можливість браузеру обмінюватись даними з сервером без перезавантаження сторінки, що робить використання сервісу більш швидким та зручним.

2. ОГЛЯД ПРОБЛЕМАТИКИ МОДЕЛЮВАННЯ 3D ОБ'ЄКТІВ

2.1 Проблематика прототипування

Виготовленням прототипів та 3D друком називають процес створення, так званого технічного зразка продукції на основі готової 3D моделі за допомогою спеціальних друкованих пристроїв, якими можна управляти за допомогою комп'ютера. Принтер виробляє предмет починаючи з самого нижнього шару, після висихання якого починається накладання наступного шару і так далі, доки не буде отримана потрібна об'ємна конструкція [3].

Створення дослідних зразків за допомогою 3D друку значно скорочує час і витрати виробництва, а завдяки можливостям 3D моделювання спектр проектованих деталей практично не обмежений. Прототипування дозволяє наочно оцінити можливі недоліки виробу ще на етапі проектування і внести істотні зміни в конструкцію деталі ще до її остаточного затвердження, при цьому витративши мінімальну кількість матеріалів та ресурсів [4].

Найбільш поширеними програмами для прототипування є Blender, SketchUp Make та Autodesk 123D. Програми містять інструменти анімації, моделювання, рендеринга, а також засоби для створення відео.

Однак, головним недоліком десктопних програм є необхідність їх інсталяції та постійного оновлення, що безумовно збільшує поріг входження до 3D моделювання.

2.2 Проблематика хмарних сервісів

З появою хмарних технологій дедалі більшу популярність набувають і хмарні сервіси швидкого прототипування, що дозволяють з будь-якої точки світу синхронізувати дані між персональним пристроєм та надійним місцем зберігання.

Хмарні сервіси на відмінну від звичайних програм для прототипування мають такі переваги [5]:

1. Немає необхідності витрачати кошти на придбання дорогих та потужних комп’ютерів для інсталяції програмного забезпечення.

2. Користувач не прив’язаний до робочого місця, де встановлена необхідна програма, достатньо просто мати доступ до інтернету.

3. Усі інструменти необхідні для роботи з 3D моделями надаються веб-сервісами.

4. Високий рівень обчислювальних потужностей, який надається користувачу, дозволяє зберігати, аналізувати і обробляти дані з максимальною швидкістю та надійністю.

Однак використання хмарних сервісі досі не витіснило десктопні програми, завдяки своїм недолікам, це:

1. Наявність постійного високошвидкісного та якісного Інтернету.

2. Деякі операції, пов'язані з великими обсягами передачі інформації можуть виконуватися повільніше, ніж за допомогою програми встановленої на персональному комп'ютері.

3. Надійність клієнтських даних може бути під загрозою, якщо сервіс погано шифрує дані або вразливий до вторгнення шкідливого програмного забезпечення.

2.3 Реалізація сервісу швидкого прототипування

Метою роботи було створення власного хмарного сервісу швидкого прототипування, а також обчислення об’єму завантаженої STL моделі, її маси та витрат матеріалу для друку.

Сам сервіс реалізовано на програмній платформі Node.js з використанням технології Ajax. У якості інструментів для написання клієнтської частини було використано мову розмітки HTML, спеціальну мову для опису зовнішнього виду сторінок CSS, мову програмування JavaScript, а для роботи з 3D моделями у браузері – JavaScript бібліотеки Three.js, STLLoader.js та TrackballControls.js.

Особливістю даного сервісу є те, що він написаний на основі архітектурного стилю REST, який повинен відповідати таким вимогам [6]:

1. Модель «клієнт-сервер». Відділення потреби інтерфейсу клієнта від потреб сервера, що зберігає дані, підвищує переносимість коду клієнтського інтерфейсу на інші платформи, а спрощення серверної частини покращує масштабованість.

2. Відсутність стану. В період між запитами клієнта ніяка інформація про його стан на сервері не зберігається. Всі запити від клієнта повинні бути складені так, щоб сервер отримав всю необхідну інформацію для виконання запиту.

3. Кешування. Клієнти можуть виконувати кешування відповідей сервера. Відповіді сервера, в свою чергу, повинні мати явне або неявне позначення як кешувального або не кешувального з метою запобігання отримання клієнтами застарілих або невірних даних у відповідь на наступні запити.

4. Одноманітність інтерфейсу. Така вимога до інтерфейсу дозволяє кожному з сервісів незалежно розвиватися.

5. Можливість застосовувати проміжні сервери. Дана особливість дозволяє підвищити масштабованість за рахунок балансування навантаження і розподіленого кешування.

6. Можливість завантаження коду з сервера у вигляді аплетів. Це у свою чергу дозволить підвищити функціональність клієнта.

2.4 Висновки до розділу

У даному розділі було розглянуто проблематику прототипування та хмарних сервісів, а також описано засоби для реалізації власного хмарного сервісу та його особливості.

3. ОГЛЯД ТЕХНОЛОГІЙ ПРОГРАМУВАННЯ ДЛЯ РЕАЛІЗАЦІЇ КЛІЄНТ-СЕРВЕРНОЇ АРХІТЕКТУРИ ХМАРНИХ РІШЕНЬ

3.1 Критерії вибору технологій

На сьогоднішній день індустрія розробки онлайн-сервісів досить стрімко розвивається і перед розробниками стоїть питання між вибором технологій програмування, адже вони мають величезне значення, накладаючи відбиток на роботу всього додатку або системи.

При розробці хмарного сервісу в першу чергу потрібно вибрати надійний та зручний інструмент для написання REST серверу. Найбільш популярними серверними мовами на даний момент є PHP та Node.js. Однак вибір на користь Node.js було зроблено на основі таких критеріїв:

1. Зручність написання коду.
2. Наявність документації та інформаційних ресурсів.
3. Наявність багатофункціональних вбудованих інструментів для розробники.
4. Швидкий інтерпретатор.
5. Висока швидкість виконання запитів.

3.2 Аналіз технологій для створення сервісу швидкого прототипування

3.2.1 Переваги та недоліки Node.js при створені серверу

Програмна платформа Node.js – це представник нових технологій веб-розробки [7]. На відміну від PHP Node.js не є мовою програмування – це середовище

виконання, яке використовує мову JavaScript для написання додатків на стороні сервера.

Серед основних переваг використання Node.js можна виділити такі [8]:

1. Швидке серверне рішення. Середовище Node.js дозволяє використовуючи чергу подій JavaScript створювати додатки з не блокуючим введенням-виведенням, які здатні обробляти декілька запитів одночасно, що є досить суттєвим для майбутнього REST серверу до якого буде надходити велика кількість запитів клієнтів.

2. Одна мова для Front-end та Back-end. Використовуючи Node.js на сервері можна отримати всі переваги скриптової мови JavaScript як на стороні клієнта, так і на стороні сервера.

3. Гнучкість. У середовищі Node.js не має жорстких правил або залежностей, що звільняє простір для створення різноманітних додатків. Розробники самі вибирають архітектуру залежностей.

4. Велика кількість зовнішніх бібліотек та готових модулів. Програмна платформа Node.js має багатофункціональний та зручний пакетний менеджер npm.

5. Використання JSON формату. Даний формат є найбільш використовуваним та досить зручним для обміну даними в Інтернеті.

Однак дане середовище має і деякі недоліки, зокрема:

1. Високий поріг входження.

2. Не доцільність використання для масивних обчислень.

3. Проблема з розміщенням на хостингу. Оскільки для Node.js потрібна віртуальне хмара (VDS / VPS, серверна середовище, з повним доступом). На жаль, таке можуть собі дозволити не всі хостери, тому і ціни будуть відповідні.

3.2.2 Переваги та недоліки Ajax

У якості технології для обміну інформацією з сервером було обрано Ajax, адже її використання дозволяє покращити функціональність та зовнішній вигляд сторінки користувача. Крім того, Ajax робить запити без перезавантаження сторінки

в результаті чого додаток стає більш зручним та швидким для користувача [9].

Основними перевагами Ajax є:

1. Економія Інтернет-трафіку користувача (адже замість повного оновлення сторінки відбувається перезавантаження лише невеликої зміненої частини).

2. Зменшення навантаження на сервер.

3. Більш швидка реакція інтерфейсу на команди користувача.

Однак Ajax має і ряд недоліків, зокрема [10]:

1. Контент, який завантажується динамічно не доступний пошуковим системам.

2. В браузері користувача повинен бути включений JavaScript.

3. Факт відсутності перезавантаження веб-сторінок в процесі інтеракцій призводить до неможливості збереження історії сесії.

4. Використання JavaScript може створити проблеми з закладками. Так як динамічно сторінки генерує JavaScript, а не сервер, то адреса сторінки вирізається з циклу і вже ніяк не може бути використана у подальшому.

3.3 Переваги та недоліки використання бібліотеки Three.js

Окрім розрахунку витрат матеріалу для друку, розроблювальний сервіс матиме також окреме вікно, де користувач зможе оглянути завантажену модель для оцінки її ергономічності та дизайну.

Реалізувати дану задачу дозволяє JavaScript бібліотека Three.js [11] – це бібліотека, використання якої дозволяє працювати з 3D графікою при розробці веб-додатків.

Основними перевагами Three.js є:

1. Легке підключення та доступне використання.

2. Можливість відображення і маніпулювання тривимірною графікою на веб-сторінках за допомогою мови JavaScript.

3. Можливість додавання та видалення 3D об'єктів в режимі реального часу.

4. Кросплатформеність. Користувачі можуть побачити 3D модель на будь-

яких платформах, адже при перегляді їх об’єднує браузер.

5. Можливість працювати з інтерфейсним елементом canvas, завдяки чому 3D модель може відображатись і на багатьох мобільних пристроях.

До недоліків можна віднести:

1. Для відображення 3D об’єктів у браузері необхідно створити сцену, камеру та візуалізатор, що є мало зрозумілим для звичайного розробника.

2. Підтримка лише у сучасних браузерах.

3.4 Висновки до розділу

У даному розділі було розглянуто технології для реалізації клієнт-серверної архітектури хмарних рішень, проведено порівняння з основними аналогічними технологіями та обґрунтовано їх вибір.

Не зважаючи на всі недоліки Node.js, Ajax та бібліотеки Three.js для реалізації хмарного сервісу швидкого прототипування ці технології є найкращим рішенням, адже вони є найбільш підходящими для створення REST архітектури та роботою з 3D моделями у форматі STL з найбільшою швидкістю та найменш затраченими ресурсами.

4. НАЛАШТУВАННЯ ПЛАТФОРМИ NODE.JS ТА БІБЛІОТЕК НА СТОРОНІ КЛІЄНТА

4.1 Підготовка середовища Node.js та особливості його використання

Для початку роботи з середовищем Node.js його необхідно встановити на робочий комп'ютер, це можна зробити з офіційного сайту за таким посиланням https://nodejs.org/en/.

Даний файл для інсталяції відповідає 64 розрядній операційній системі Windows. Після чого завантажуємо файл натиснувши на «12.2.0 Current» (рисунок 4.1).

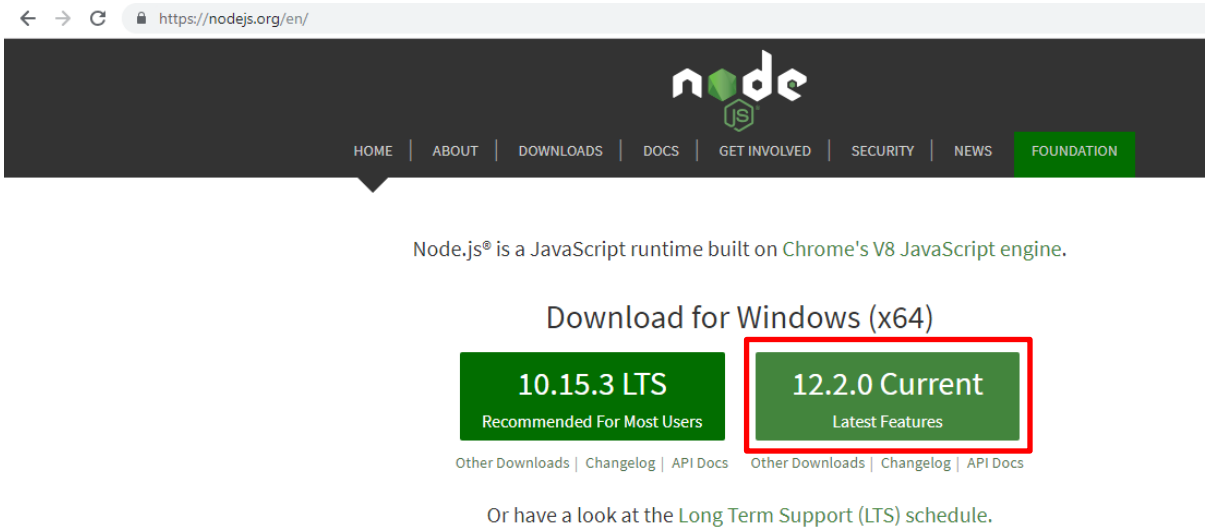

Рисунок 4.1 – Вибір файлу для завантаження Node.js

Після того як файл завантажився, відкриваємо його та натискаємо на кнопку «Запустити» («Запустить») (рисунок 4.2).

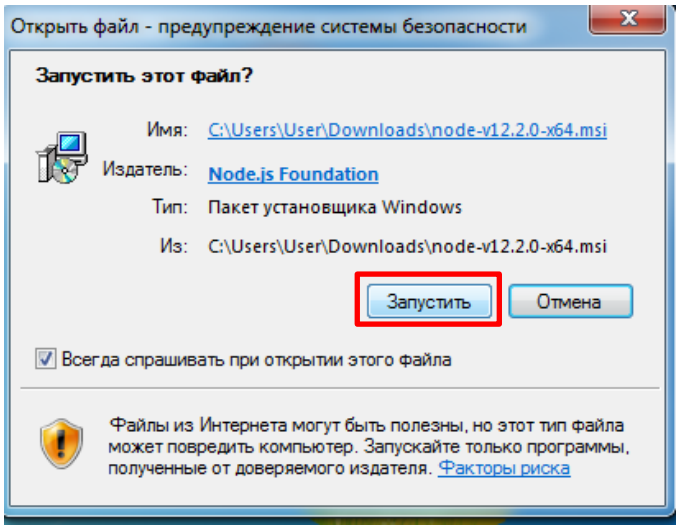

Рисунок 4.2 – Запуск процесу інсталяції

Далі переходимо до самого процесу встановлення натиснувши на кнопку «Далі» («Next») (рисунок 4.3).

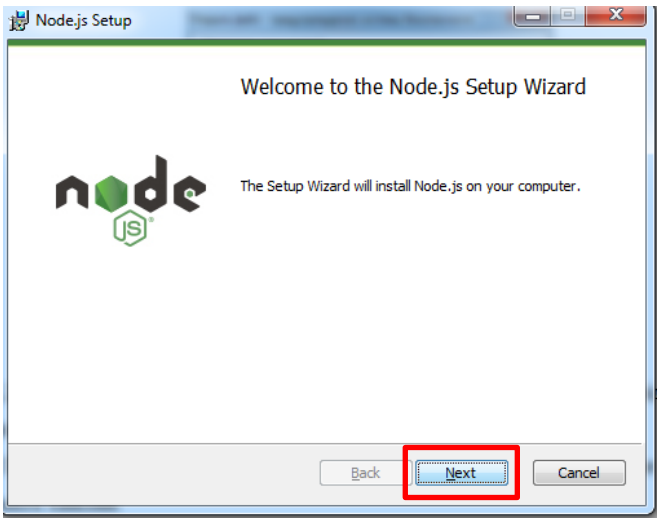

Рисунок 4.3 – Погодження з процесом інсталяції

Ознайомлюємось з правилами (ставимо галочку) та натискаємо на кнопку «Далі» («Next») (рисунок 4.4).

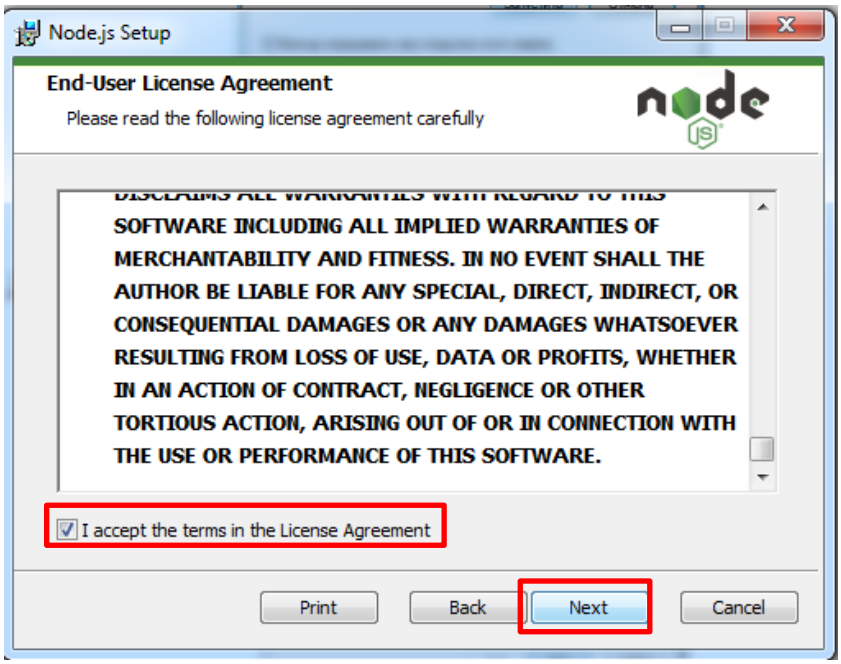

Рисунок 4.4 – Ознайомлення з правилами

Далі вибираємо місце встановлення програми (рисунок 4.5) та необхідні компоненти (рисунок 4.6) та натискаємо «Далі» («Next»).

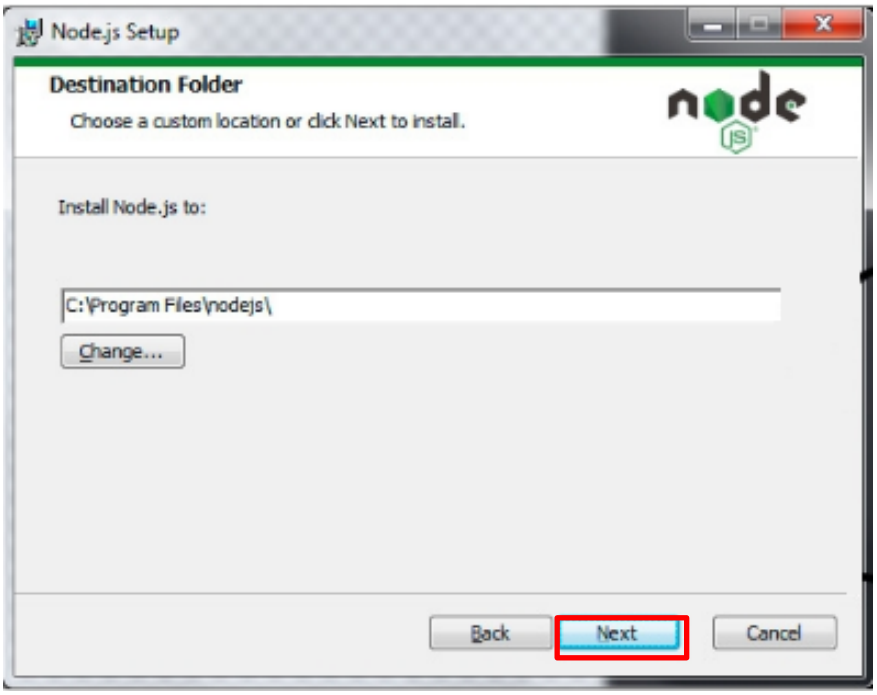

Рисунок 4.5 – Вибір місця встановлення програми

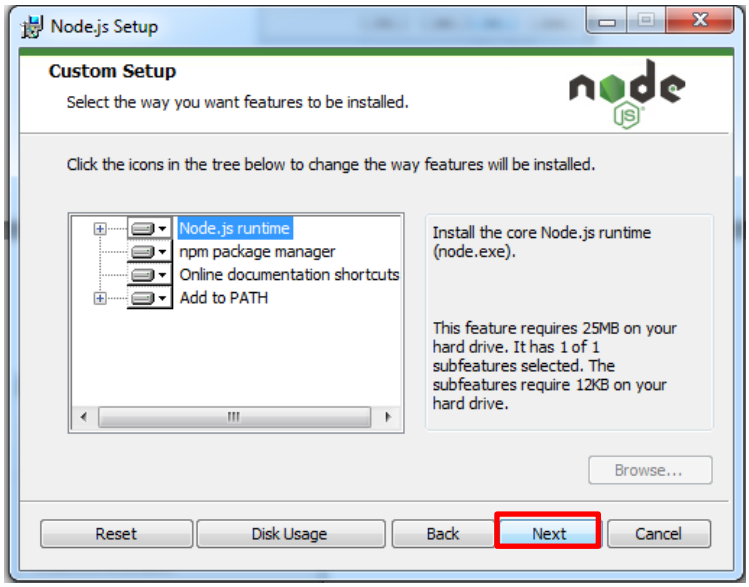

Рисунок 4.6 – Вибір необхідних компонентів програми

Починаємо процес інсталяції натиснувши на «Завантажити» («Install») (рисунок 4.7).

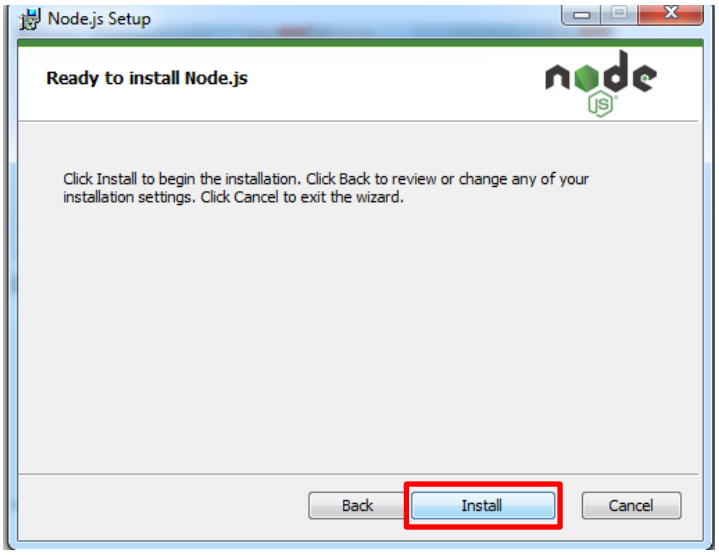

Рисунок 4.7 – Початок процесу інсталяції

Після декілька хвилиного процесу інсталяції Node.js буде встановлений та з’явиться вікно, де потрібно натиснути на кнопку «Завершити» («Finish») (рисунок 4.8).

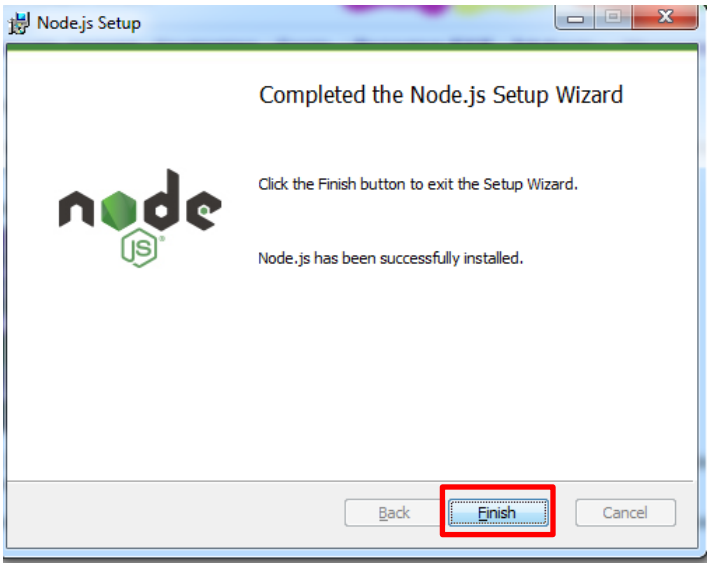

Рисунок 4.8 – Завершення інсталяції

Для перевірки інсталяції Node.js та пакетного менеджер npm, потрібно відкрити командний рядок та ввести наступні команди після чого повинна з'явитися версія Node.js та версія npm (рисунок 4.9).

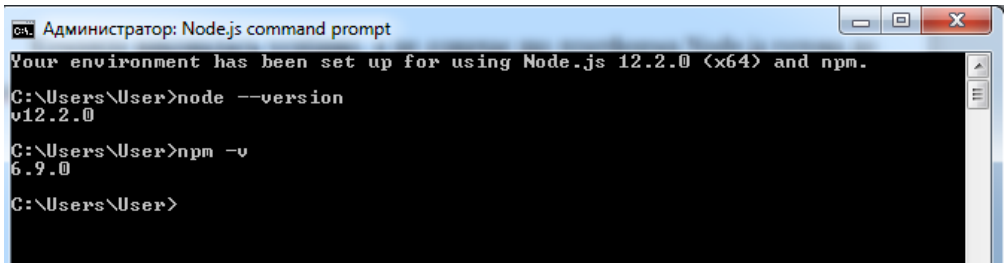

Рисунок 4.9 – Перевірка встановлення Node.js

Команди виконались успішно, а це означає що платформа Node.js та пакетний менеджер готові до використання.

4.2 Підключення 3D бібліотек та особливості їх використання

Для роботи з 3D графікою у браузері необхідно підключити такі бібліотеки:

— three.js – для відображення 3D графіки у веб-браузерах;

— STLLoader.js – для завантаження STL файлів у веб-браузер;

— TrackballControls.js – для можливості змінювати розташування завантаженої моделі за допомогою комп’ютерної мишки.

Для підключення бібліотеки Three.js необхідно перейти на офіційний сайт, який знаходиться за наступним посиланням https://threejs.org/ та перейти у вкладку «download» після чого почнеться завантаження архіву з бібліотекою (рисунок 4.10).

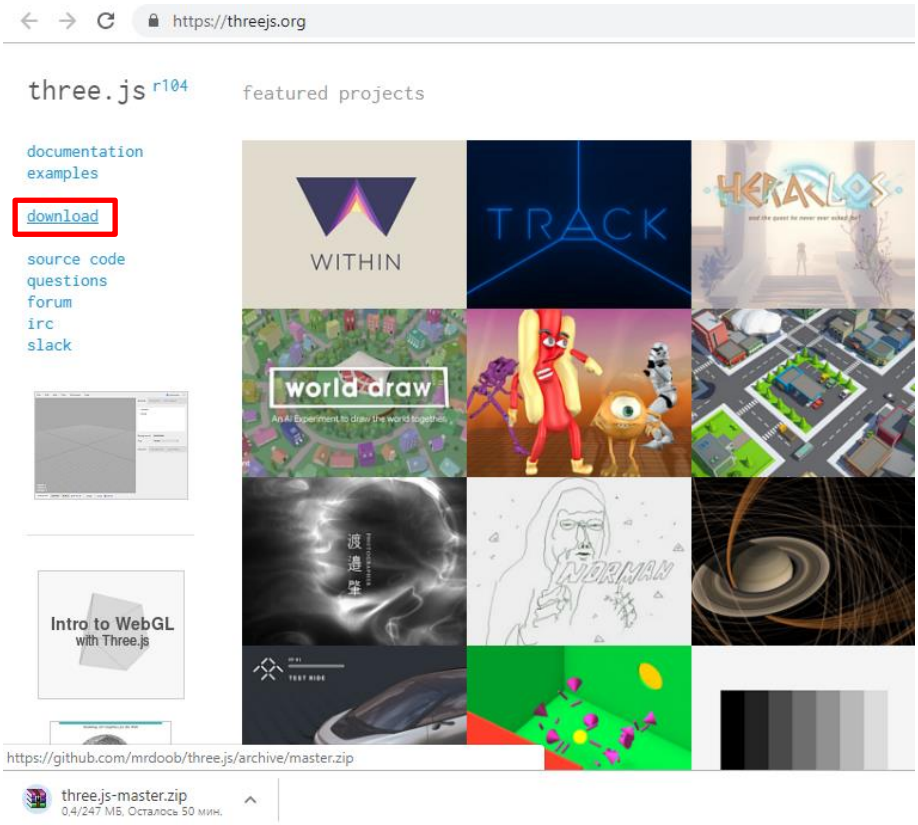

Рисунок 4.10 – Завантаження бібліотеки Three.js

Розпакувавши архів необхідно зайти у папку build та вилучити файл three.min.js до свого проекту (рисунок 4.11).

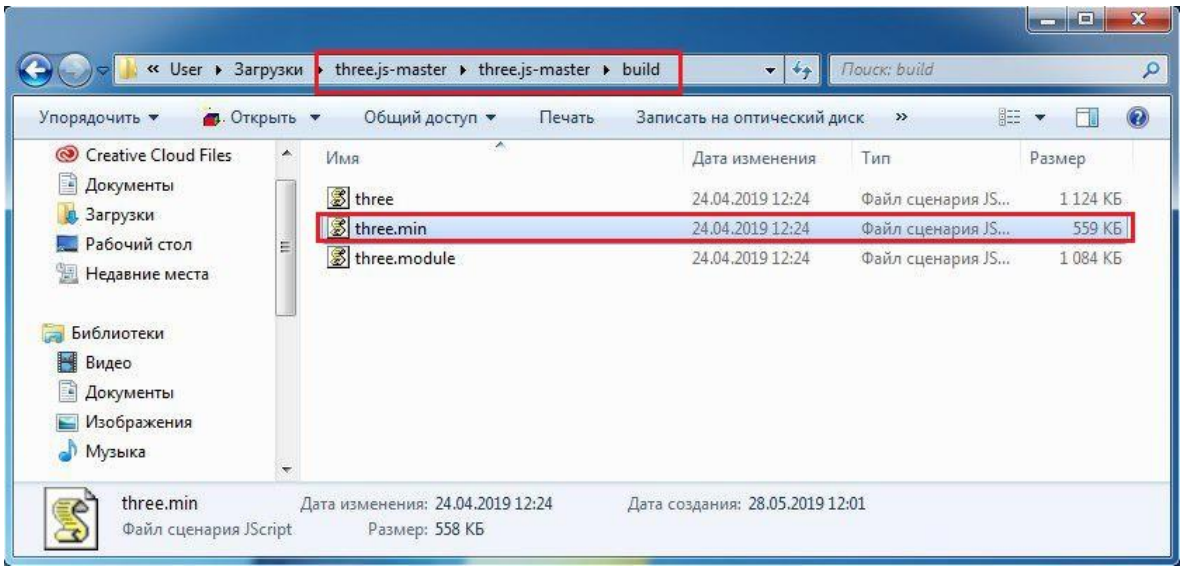

Рисунок 4.11 – Вилучення необідного файлу three.js до проекту

Для підключення бібліотеки STLLoader.js потрібно перейти у вкладку «examples» на у пошуковому рядку ввести початок назви бібліотеки, після чого з’явиться посилання на потрібну бібліотеку «loader / stl». Перейшовши на вкладку відкриється вікно в правому нижньому кутку якого буде посилання на веб-сервіс GitHub («View source»), де знаходиться код бібліотеки (рисунок 4.12).

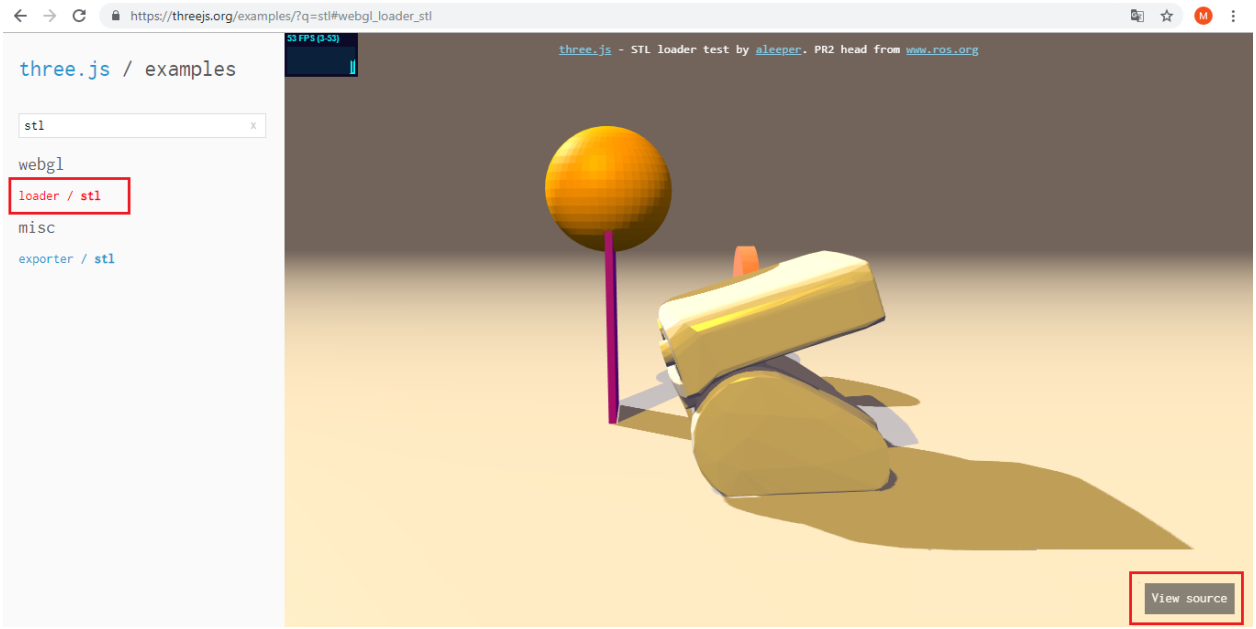

Рисунок 4.12 – Підключення бібліотеки STLLoader.js

Перейшовши в папку loaders в ній знаходиться потрібний файл який потрібно

завантажити до проекту (рисунок 4.13).

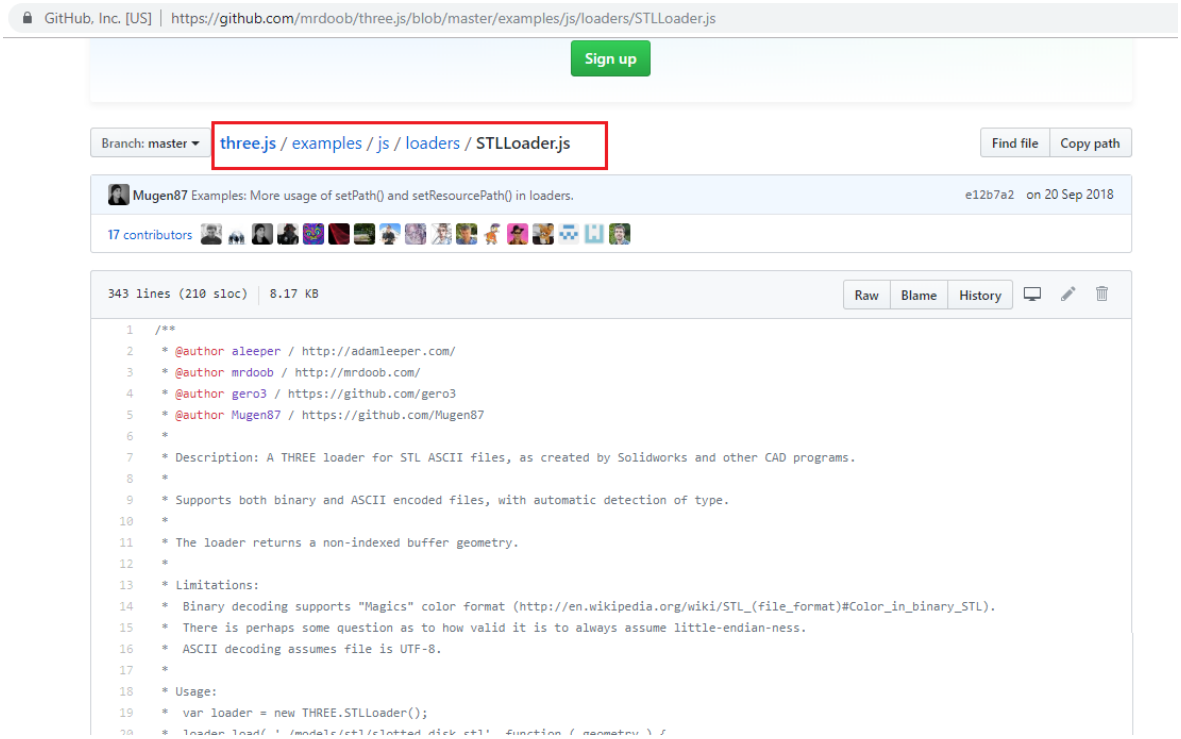

Рисунок 4.13 – Файл STLLoader.js

Завантажити бібліотеку TrackballControls.js можна теж через вкладку «examples» (рисунок 4.14), через яку можна отримати посилання на GitHub.

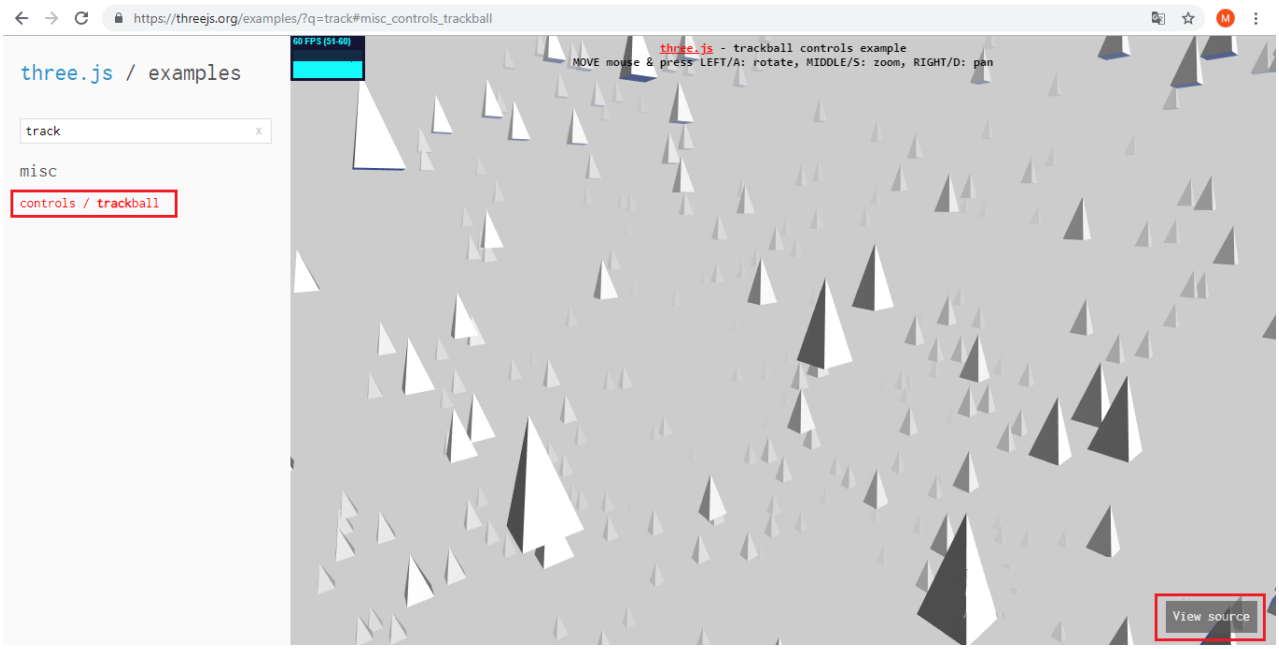

Рисунок 4.14 – Підключення бібліотеки TrackballControls.js

У папці controls знаходяться всі можливі варіанти курування моделлю, до проекту потрібно завантажити саме файл TrackballControls.js (рисунок 4.15).

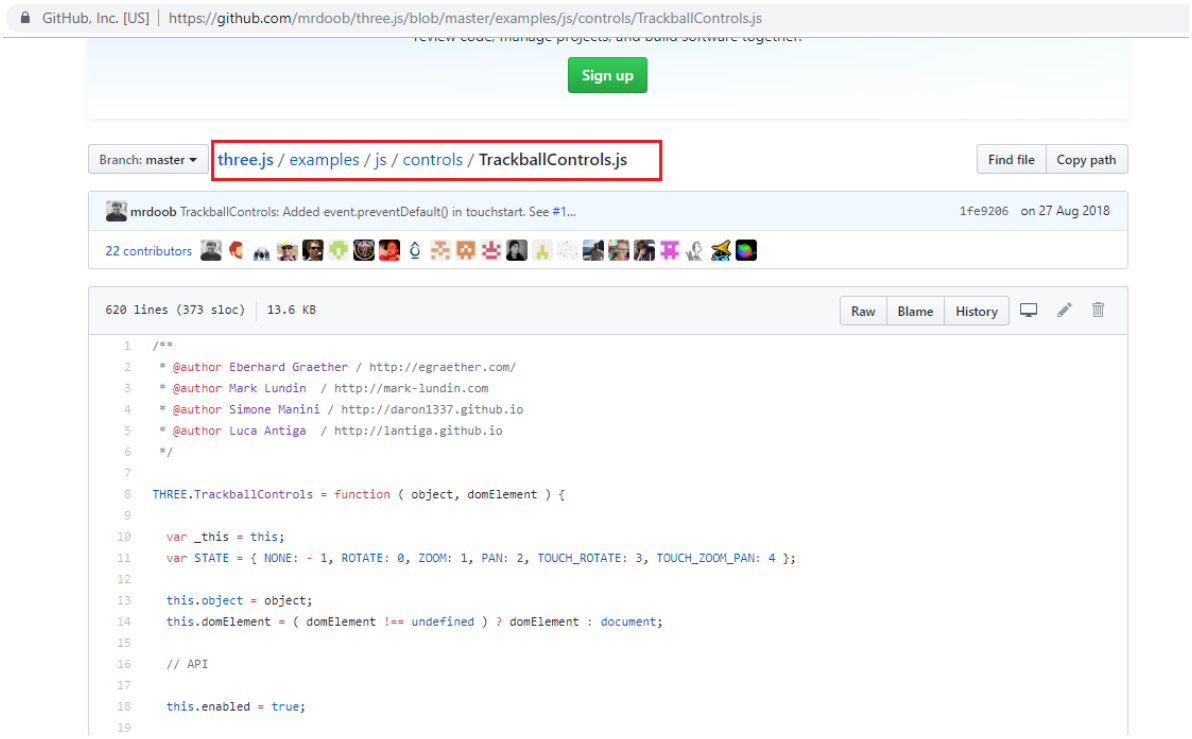

Рисунок 4.15 – Файл TrackballControls.js

Отримавши всі JavaScript файли їх необхідно підключити перед закриваючим тегом head в файлі index.html у такій послідовності: Three.js, STLLoader.js, TrackballControls.js.

4.3 Висновки до розділу

У даному розділі було описано етапи інсталяції платформи Node.js та клієнтських бібліотек для роботи з 3D графікою у браузерах – Three.js, STLLoader.js та TrackballControls.js.

5. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ 3D ПРОТОТИПУВАННЯ

5.1 Опис функціональності системи

Можливості даної системи можна побачити на діаграмі прецендентів (use case діаграмі) (рисунок 5.1).

Рисунок 5.1 – Діаграма прецендентів

Діграма прецендентів описує відношення між акторами та прецендентами в системі. Суть даної діаграми полягає в тому, що проектована система представлена у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою

так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою [12].

5.2 Створення взаємодії серверу з клієнтським додатком

Для реалізації клієнт-серверної архітектури необхідно створити файл index.html (клієнтська частина) та server.js (сам сервер).

У файлі index.html перед закриваючим тегом head необхідно виконати підключення таких файлів:

— бібліотеки Three.js;

— бібліотеки STLLoader.js;

— бібліотеки TrackballControls.js;

— бібліотеки jQuery;

— бібліотеки jQuery Form;

— JavaScript файлу зі сценою, камерою та візуалізатором.

Для завантаження клієнтської моделі на сервер необхідно створити форму з атрибутами enctype = «multipart/form-data» та method = «post», а також полем для вибору користувачем файлу та кнопкою відправки (рисунок 5.2).

```
<form id="frmUploader" enctype="multipart/form-data" action="/calculate" method="post">
  <input type="file" id="file" name="file" multiple>
  <label for="file" class="uploadButton">Вибрати модель</label>
  <input type="submit" name="submit" id="btnSubmit" value="Розрахувати вартість">
</form>
```

Рисунок 5.2 – Форма відправки файлу

У файлі server.js знаходиться сам сервер, для повноцінної роботи якого потрібно підключити такі пакетні модулі:

— express;

— path;

— multer;

— cors;

— body-parser;

— parse-stl;

— fs.

Для роботи даної системи було реалізовано два запити: GET – який повертає файл index.html для початку роботи та POST – який передає STL модель на сервер, де проводяться необхідні обчислення, та повертає їх на сторону клієнта (рисунок 5.3).

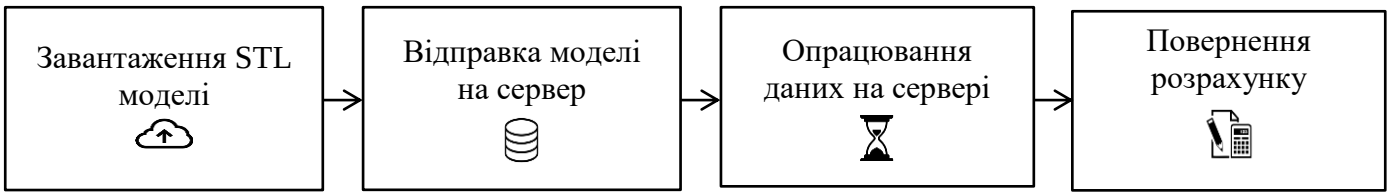

Рисунок 5.3 – Схема реалізації клієнт-серверної архітектури

Оскільки всі розрахунки проводяться на стороні сервера, то модель, яку завантажує користувач повинна знаходитись на сервер. Для цього на ньому потрібно виділити місце куди буде завантажено цей файл для подальших обчислень (рисунок 5.4).

```
var Storage = multer.diskStorage({
  destination: function (req, file, callback) {
    callback(null, "./models");
  },
  filename: function (req, file, callback) {
    callback(null, file.originalname);
  }
});

var upload = multer({ storage: Storage }).array("file", 3);
```

Рисунок 5.4 – Створення сховища для моделей

Таким чином, STL моделі, які завантажує користувач будуть надсилатись на сервер у папку models з тим ім’ям, яке мали при завантажені.

5.3 Реалізація алгоритму пересилання даних за допомогою Ajax

Для обміну данними з сервером було використано технологію Ajax, яка використовує пакет multer [13] – це проміжне програмне забезпечення Node.js для обробки багаточастинних / форм-даних, які в основному використовуються для завантаження файлів.

Для того, аби дані завантажувались на сервер, необхідно відстежити подію натиснення на кнопку відправки форми та використовуючи конструкцію fetch [14] передати URL адресу сторінки (рисунок 5.5), яка повинна збігатись з каталогом POST запиту на сервері (рисунок 5.6).

```
const url = 'http://localhost:3000/calculate';
const btn = document.getElementById('btnSubmit');
var formatFile;

btn.onclick = function (e) {

    const file = document.getElementById('file').files[0];
    var formatFile = file.name.split(".").splice(-1,1)[0];

    // перевірка формату файла (тільки STL)
    if (formatFile.toLowerCase() != 'stl') {
        document.getElementById("error").innerHTML = "Помилка формату! Ваш файл не STL ";
        e.preventDefault();
    // перевірка на підтримку браузера API файлів
    } else if (window.File && window.FileReader && window.FileList && window.Blob) {
        const formData = new FormData();
        formData.append("file", file);

        // запит на сервер по URL
        fetch(url, {
            method: 'post',
            qwerty: formData,
            body: formData,
        }).then(function (response) {
            // відповідь з сервера
            console.log(response);

            response.json().then(e => {
                console.log(e);
```

Рисунок 5.5 – Запит на сервер по вказаній URL адресі

```
app.post("/calculate", (req, res) => {
  upload(req, res, function (err) {
    const fileName = req.files[0].originalname;
    if (err) {
      console.log("Ошибка: " + err);
      return res.send({ error: "Something went wrong!" });
    }
    const file = __dirname + '/models/' + fileName;
    var buf = fs.readFileSync(file);
    var mesh = parseSTL(buf);
    var pos = mesh.positions;

    if (pos.length % 3 != 0) {
        return res.send({
            success: false, message: "Error!"
        });
    };

    var model_volum = calcVolum(pos);

    var density = 1.04;
    var model_weight = model_volum * density;
    var length_strings = model_weight / 3;
    var unit_price = 2.25;
    var cost_model = length_strings * unit_price;

    return res.send({
      success: true, name: fileName,
      volum: model_volum, weight: model_weight,
      length: length_strings, cost: cost_model
    });
  });
});
```

Рисунок 5.6 – POST запит на сервер

Таким чином, для того аби здійснити запит на сервер та завантажити на нього модель за допомогою технології Ajax необхідно виконати наступні вимоги [15]:

1. У формі повинно бути вказано два атрибути enctype = «multipart/form-data» та method = «post».

2. У поля для завантаження файлу повинен бути атрибут multiple.

3. На стороні сервера необхідно підключити пакет multer.

4. На сервері повинно бути створене сховище для моделей.

5. URL адреса конструкції fetch та каталог запиту на сервері повинні збігатися.

5.4 Реалізація алгоритму розрахунку витрат матеріалу для 3D друку

У даній роботі всі розрахунки відбуваються з урахуванням того, що модель є суцільною, а її друк виконується найпопулярнішим видом пластика – ABS. Тому, головною метою було обчислення таких величин:

— об’єму моделі;

— її маси;

— довжини нитки для її друку;

— приблизну вартість друку.

Перш за все, для того аби мати можливість обчислити об’єм моделі, необхідно отримати усі її вершини. А це можливо зробити завдяки пакету parse-stl [16], який зчитує весь STL файл та повертає масив точок (рисунок 5.7).

```
const file = __dirname + '/models/' + fileName;
var buf = fs.readFileSync(file);
var mesh = parseSTL(buf);
var pos = mesh.positions;
```

Рисунок 5.7 – Повернення масиву вершин STL моделі

Таким чином, маючи масив усіх вершин трикутників моделі їх можна розглянути як тетраедр [17] (рисунок 5.8) та застосувати наступну формулу для обчислення об’єму (5.1):

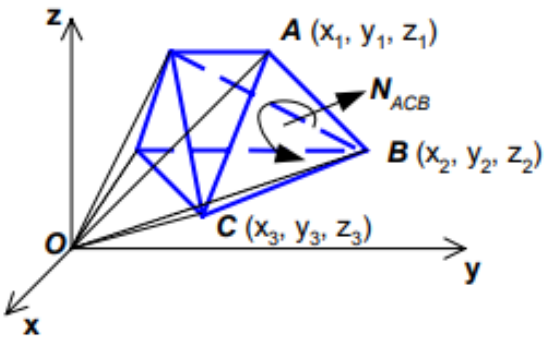

Рисунок 5.8 – Представлення трикутників у вигляді тетраедрів

$$V_i = \left| \frac{1}{6} \cdot (-x_3 \cdot y_2 \cdot z_1 + x_2 \cdot y_3 \cdot z_1 + x_3 \cdot y_1 \cdot z_2 \right. \\ \left. -x_1 \cdot y_3 \cdot z_2 - x_2 \cdot y_1 \cdot z_3 + x_1 \cdot y_2 \cdot z_3) \right|, \quad (5.1)$$

де (x_i, y_i, z_i) – координати і-тої точки.

Загальний об’єм моделі – це сума об’ємів тетраедрів з яких вона складається (5.2).

$$V = \sum_i V_i \quad (5.2)$$

Знаючи об'єм моделі та враховуючи, що середня густина ABS пластика $1{,}04\ \frac{\text{кг}}{\text{м}^3}$ (ρ) можна знайти і її масу за наступною формулою (5.3):

$$m = p \cdot V \quad (5.3)$$

Для обчислення довжини нитки [18], яку необхідно використати для друку моделі можна скористатись такою формулою (5.4):

$$l = \frac{m}{3} \quad (5.4)$$

Приблизна вартість кілограмової котушки ABS пластику – 900 грн, а довжина нитки в котушці – 400 м. Тобто, ціна одного метру пластику (C_1) – 2,25 грн.

Приблизну вартість друку (C) можна обчислити за наступною формулою (5.5):

$$\mathrm{C} = \mathrm{C}_1 . l \quad (5.5)$$

5.5 Повернення результату розрахунків на сторону клієнта

Провівши усі необхідні обчислення на стороні серверу необхідно повернути ці значення для відображення клієнту. Найбільш зручний спосіб це зробити – повернути данні у форматі JSON [19] (рисунок 5.9).

```
return res.send({
  success: true,
  name: fileName,
  volum: model_volum,
  weight: model_weight,
  length: length_strings,
  cost: cost_model
});
```

Рисунок 5.9 – Повернення обчислень з серверу у форматі JSON

Отримати ж дані на стороні клієнта можна за допомогою об’єкту response [20] в конструкції fetch яка і здійснює запит (рисунок 5.10).

```
// запит на сервер по URL
fetch(url, {
    method: 'post',
    qwerty: formData,
    body: formData,
}).then(function (response) {
    // відповідь с сервера
    console.log(response);

    response.json().then(e => {
    console.log(e);
```

Рисунок 5.10 – Отримання обчислень на стороні клієнта

Таким чином, технологія Node.js дає можливість досить зручно передавати дані з сервера, що безумовно підвищує ефективність роботи з сервісом як для розробників, так і для самих користувачів.

5.6 Висновки до розділу

У даному розділі було описано функціональність системи, реалізацію серверу з клієнтом, розглянуто особливості роботи з Ajax при завантажені даних на сервер, а також описано алгоритму розрахунку витрат матеріалу для 3D друку.

6. ВИПРОБУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

Сервіс було розроблено з акцентом на простоту та зручність у використанні клієнтом, а також на швидкість обробки даних. Основною відмінністю від аналогів є те, що у ньому є вікно для відображення завантаженої моделі, а також можливість її розглянути ближче використовуючи лише комп'ютерну мишку.

6.1 Сценарій взаємодія користувача з системою

При відкритті головної сторінки сервісу перед користувачем відкриється вікно браузера у якому буде дві кнопки: «Вибрати модель» та «Розрахувати вартість» (рисунок 6.1).

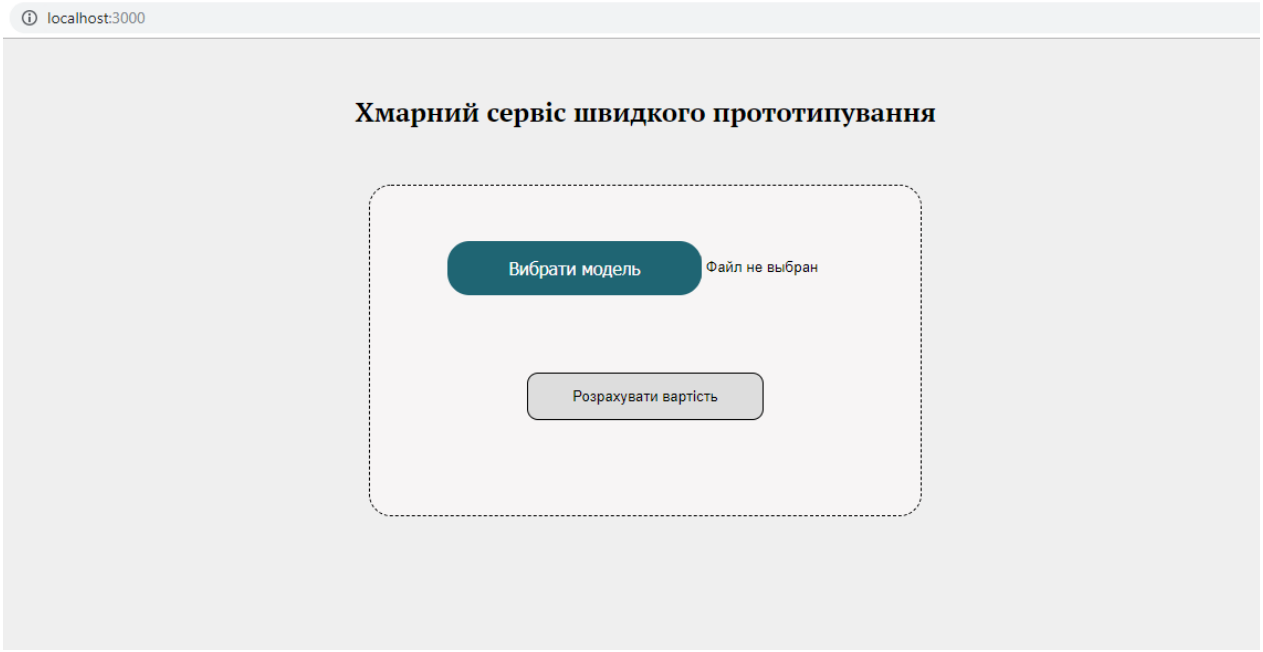

Рисунок 6.1 – Головне вікно сервісу

При натисненні на кнопку «Вибрати модель», перед користувачем відкриється вікно у якому йому необхідно вибрати модель у форматі STL, витрати на друк якої необхідно порахувати (рисунок 6.2).

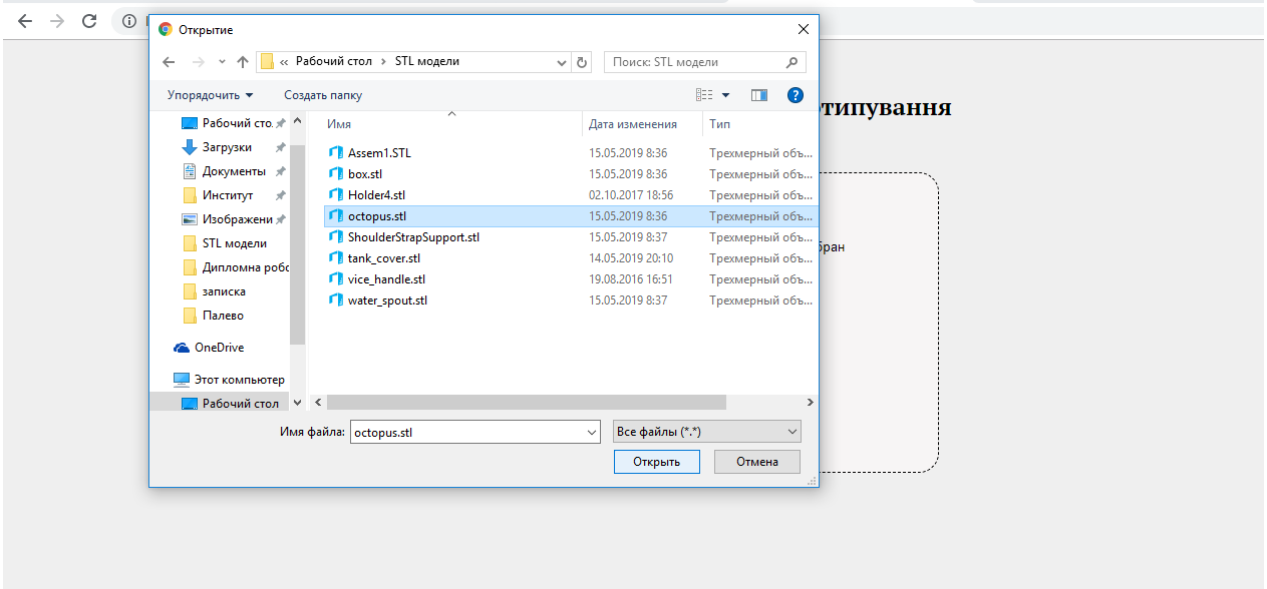

Рисунок 6.2 – Вікно вибору STL файлу

Після вибору файлу його ім’я повинно з’явитися біля кнопки «Вибрати модель», це означатиме що модель завантажена і можна провести необхідні розрахунки (рисунок 6.3).

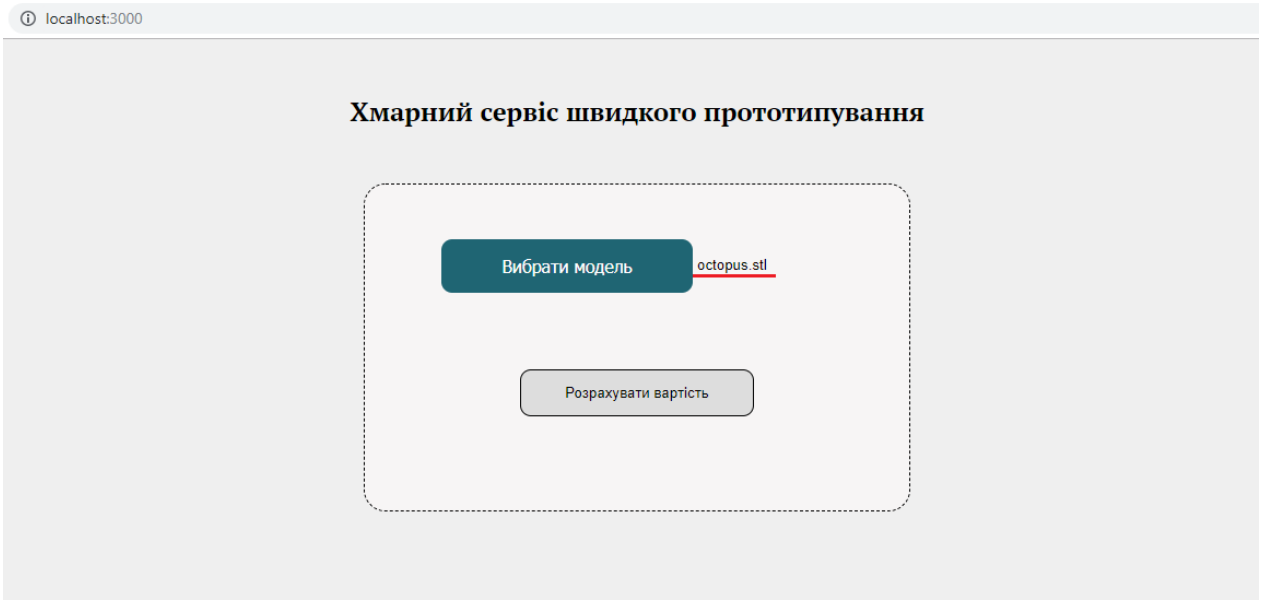

Рисунок 6.3 – Вікно з завантаженим файлом

Після того як користувач натиснув на кнопку «Розрахувати вартість» то модель буде завантажена на сервер і буде виконаний POST запит, а перед ним відкриється вікно у якому буде дві області (рисунок 6.4). У одні буде знаходитись

інформація про саму завантажену модель – ім’я файлу, об’єм моделі, маса, довжина нитки для друку, приблизна вартість та матеріал для друку. У другій області представлена сама модель, яку користувач має можливість повернути в різні сторони (за допомогою комп’ютерної миші), а також приблизити або віддалити камеру та побачити як вона виглядає (рисунок 6.5).

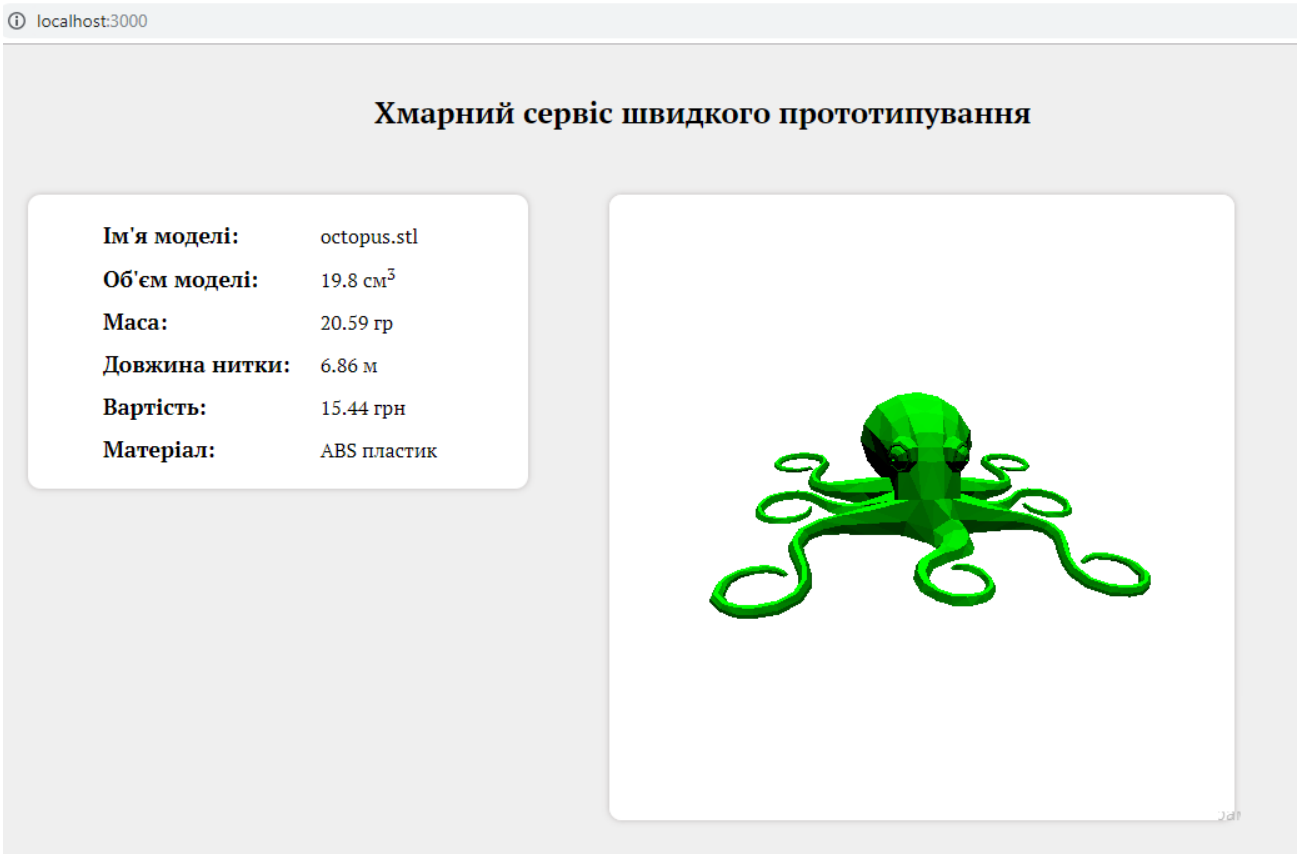

Рисунок 6.4 – Вікно з проведеними розрахунками та моделлю

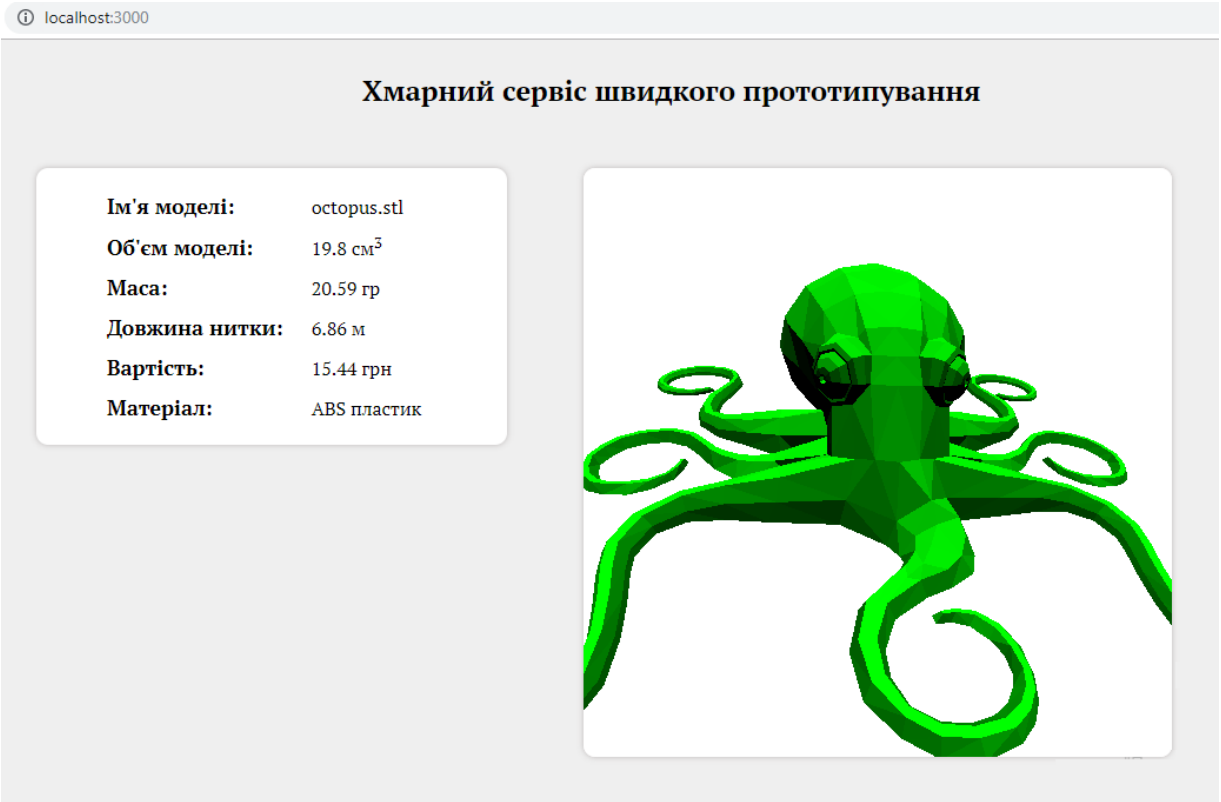

Рисунок 6.5 – Наближення камери та зміна розташування моделі

Якщо користувач вибере файл не STL формату, то програма видасть помилку (рисунок 6.6).

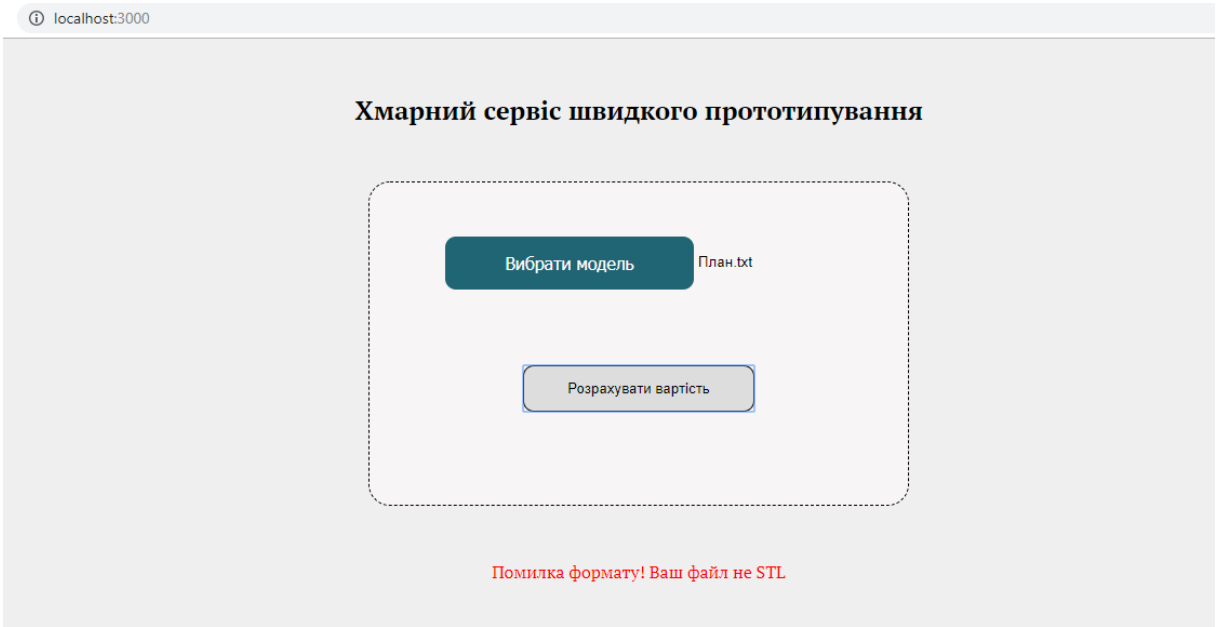

Рисунок 6.6 – Помилка при завантаженні файлу з розширенням не STL

Якщо ж модель матиме артефакти, то при натисненні на кнопку «Розрахувати вартість» програма видасть наступну помилку (рисунок 6.7).

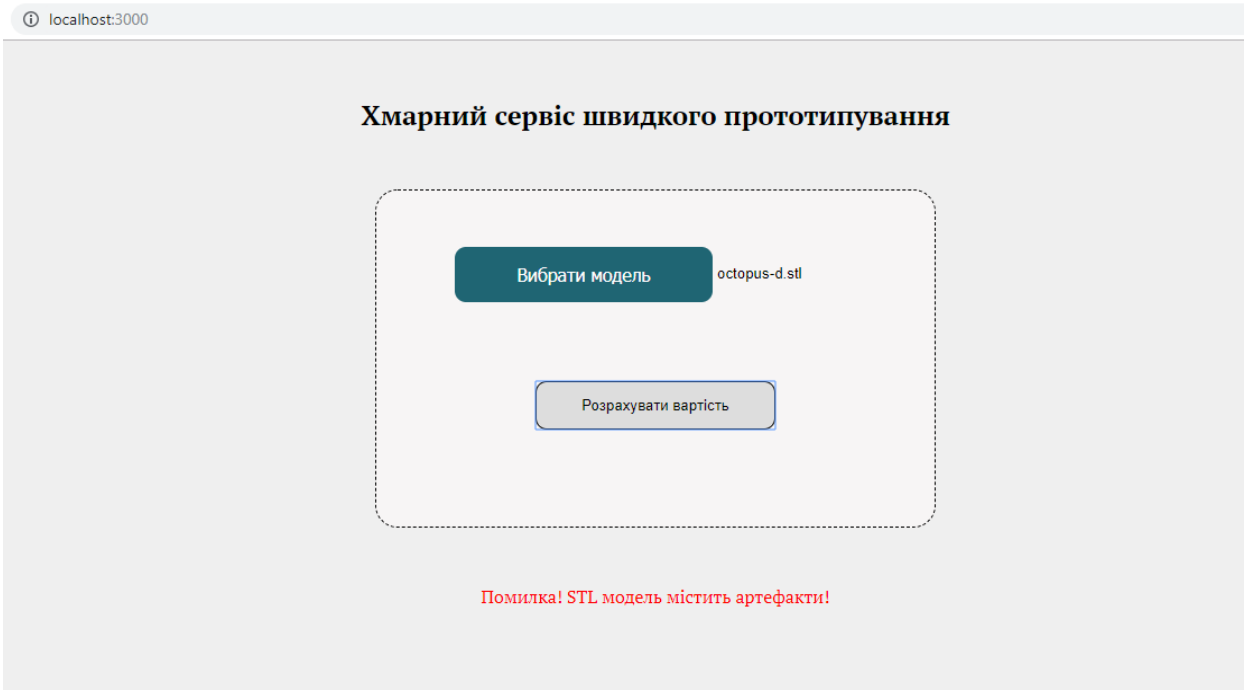

Рисунок 6.7 – Помилка, STL файл містить артефакти

6.2 Недоліки системи

Серед основних недоліків системи можна виділити такі:

1. Розрахунок приблизної вартості проводиться лише для ABS пластику.

2. У користувача є можливість лише побачити завантажену модель, але не має можливості її змінити (колір, розміри и т.д.).

3. У сервісу немає можливості задати основні параметри друку, такі як: товщина стінки або відсоток заповнення внутрішнього об’єму решітки.

6.3 Висновки до розділу

У даному розділі було продемонстровано сценарій взаємодії користувача з сервісом на конкретному прикладі, а також описано основні недоліки системи.

ВИСНОВКИ

Дана робота присвячена використанню 3D технологій в сфері 3D друку. При вирішенні поставлених задач було отримано наступні результати:

1. Аналіз проблематики прототипування та хмарних сервісів показав, що на сьогоднішній день адитивні технології все ширше використовуються для вирішення найрізноманітніших завдань, практично, у всіх областях матеріального виробництва. В той самий час використання хмарних сервісів значно зменшує поріг входження до сфери 3D моделювання.

2. Провівши порівняльний аналіз сучасних технологій програмування для реалізації клієнт-серверної архітектури було зроблено висновок, що найбільш відповідними технологіями для реалізації архітектурного стилю REST є платформа Node.js та підхід Ajax, а в якості технології візуалізації – клієнтська бібліотека Three.js. Використання зазначених технологій дозволяє створити максимально швидке серверне рішення з великою кількістю вхідних запитів.

3. Було створено хмарний сервіс швидкого прототипування, який дає можливість користувачу на основі завантаженої STL моделі розрахувати вартість її друку ABS пластиком, об’єму, маси та довжини нитки необхідної для друку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технологии 3D-прототипирования: предназначение и разновидности. [Електронний ресурс] – Режим доступу: https://3d-expo.ru/ru/article/tehnologii-3d-prototipirovaniya-prednaznachenie-i-raznovidnosti-76133.

2. Softwareon-demand, Platform as a service, Infrastructure as a service, Google Apps Education Edition. [Електронний ресурс] – Режим доступу: http://www.google.com/a/help/intl/en/edu/index.html.

3. Прототипирование. От прототипирования до производства. [Електронний ресурс] – Режим доступу: https://klona.ua/blog/3d-pechat-i-prototipirovanie/prototipirovanie-ot-prototipa-do-proizvodstva.

4. Технология прототипирования и 3D печати. [Електронний ресурс] – Режим доступу: https://3inc.ru/3d_pechat/.

5. Преимущества и недостатки облачных тенологий. [Електронний ресурс] – Режим доступу: https://sistyle.ru/blog/item/30-clouds-advantages/.

6. Erik Wilde, Cesare Pautasso. REST: From Research to Practice. — Springer Science & Business Media, 2011. — 528 p.

7. Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript = Web Development with Node and Express / Итан Браун. — Санкт-Петербург: Питер, 2017. — 336 с.

8. Дейв Крейн, Эрик Паскарелло, Даррен Джеймс. AJAX в действии: технология — Asynchronous JavaScript and XML = Ajax in Action. — М.: Вильямс, 2006. — 640 с.

9. Ajax. Принцип работы. [Електронний ресурс] – Режим доступу: https://promo.ingate.ru/seo-wikipedia/ajax/.

10. Недостатки Ajax. Не повторяйте ошибок. [Електронний ресурс] – Режим доступу: http://www.umade.ru/log/2005/06/ajax-no-more-mistakes/.

11. 3D моделирование в браузере с помощью three.js. [Електронний ресурс] – Режим доступу: https://frontender.info/building-3d-in-the-browser-with-three-js/.

12. Буч Градди. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд. / Буч Градди, Максимчук Роберт А., Энгл Майкл У., Янг Бобби Дж., Коналлен Джим, Хьюстон Келли А.: Пер с англ. – М.: ООО “И.Д. Вильямс”, 2010. – 720 с.

13. npm documentations. Multer. [Електронний ресурс] – Режим доступу: https://www.npmjs.com/package/multer.

14. Метод fetch: замена XMLHttpRequest. [Електронний ресурс] – Режим доступу: https://learn.javascript.ru/fetch.

15. Upload Files or Images to Server Using Node.js. [Електронний ресурс] – Режим доступу: https://dzone.com/articles/upload-files-or-images-to-server-using-nodejs.

16. npm documentations. Parse-stl. [Електронний ресурс] – Режим доступу: https://www.npmjs.com/package/parse-stl.

17. EFFICIENT FEATURE EXTRACTION FOR 2D/3D OBJECTS IN MESH REPRESENTATION. [Електронний ресурс] – Режим доступу: http://chenlab.ece.cornell.edu/Publication/Cha/icip01_Cha.pdf.

18. 3D принтер. Как рассчитать расход пластика и время печати. Себестоимость 3D печати. [Електронний ресурс] – Режим доступу: https://3dprinter.ua/raskhod-materialov/.

19. Использование JSON в JavaScript. [Електронний ресурс] – Режим доступу: https://www.8host.com/blog/ispolzovanie-json-v-javascript/.

20. Node.js - Response Object. [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/nodejs/nodejs_response_object.htm.

ДОДАТОК А

Хмарний сервіс швидкого прототипування

Специфікація

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б	server.js	Сервер системи
УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б	*.stl	Файл з вхідною 3D моделлю

ДОДАТОК Б

Хмарний сервіс швидкого прототипування

Текст програми

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 7

Київ 2019

Текст завантаження даних на сервер

```
$(document).ready(function () {
	var options = {
		beforeSubmit: showRequest,  // pre-submit callback
		success: showResponse  // post-submit callback
	 };

	 // bind to the form's submit event
	 $('#frmUploader').submit(function () {
		$(this).ajaxSubmit(options);
		// always return false to prevent standard browser submit and page navigation
		console.log("Form`s submit events");
		return false;
	  });
});

// pre-submit callback
function showRequest(formData, jqForm, options) {
	console.log('Uploading is starting.');
	 return true;
}

// post-submit callback
function showResponse(responseText, statusText, xhr, $form) {
	console.log('status: ' + statusText + '\n\nresponseText: \n' + responseText );
}
```

Текст розмітки сторінки

```
<h2>Хмарний сервіс швидкого прототипування</h2>
<div id="form" class="active">
	<form id="frmUploader" enctype="multipart/form-data" action="/calculate"
method="post">
		<input type="file" id="file" name="file" multiple>
		<label for="file" class="uploadButton">Вибрати модель</label>
		<input type="submit" name="submit" id="btnSubmit" value="Розрахувати вартість">
	</form>
</div>
<div id="error"></div>
<div id="info" class="hidden">
	<div class="item">
```

```
            <span>Ім'я моделі: </span>
            <span id="name-model"> </span>
        </div>
        <div class="item">
            <span>Об'єм моделі: </span>
            <span id="volum-model"> </span>
            <span>см<sup>3</sup></span>
        </div>
        <div class="item">
            <span>Маса: </span>
            <span id="weight-model"> </span>
            <span>гр</span>
        </div>
        <div class="item">
            <span>Довжина нитки: </span>
            <span id="length-strings"> </span>
            <span>м</span>
        </div>
        <div class="item">
            <span>Вартість: </span>
            <span id="cost-model"> </span>
            <span>грн</span>
        </div>
        <div class="item">
            <span>Матеріал: </span>
            <span>ABS пластик</span>
        </div>
</div>
```

Текст запиту на сервер та отримання результату

```
const url = 'http://localhost:3000/calculate';
const btn = document.getElementById('btnSubmit');
var formatFile;

btn.onclick = function (e) {

        const file = document.getElementById('file').files[0];
        var formatFile = file.name.split(".").splice(-1,1)[0];

        // перевірка формата файлу (тільки STL)
```

```
	if (formatFile.toLowerCase() != 'stl') {
		document.getElementById("error").innerHTML = "Помилка формату! Ваш файл не STL ";
		e.preventDefault();
		// перевірка на підтримку браузера API файлів
	} else if (window.File && window.FileReader && window.FileList && window.Blob) {

		const formData = new FormData();
		formData.append("file", file);

		// запит на сервер по URL
		fetch(url, {
			method: 'post',
			qwerty: formData,
			body: formData,
		}).then(function (response) {
		// відповідь з сервера
		console.log(response);

		response.json().then(e => {
			console.log(e);

			if(e.success == false) {
				document.getElementById("error").innerHTML = "Помилка! STL модель містить артефакти!";
			} else {

	// скриваємо форму, відображаємо результати + модель
	document.getElementById("error").classList.add("hidden");
	document.getElementById("form").classList.remove("active");
	document.getElementById("form").classList.add("hidden");
	document.getElementById("info").classList.remove("hidden");
	document.getElementById("info").classList.add("active");

	// відображаємо результат с сервера
	document.getElementById("name-model").innerHTML = e.name;
	document.getElementById("volum-model").innerHTML = Math.round(e.volum * 100) / 100;
	document.getElementById("weight-model").innerHTML = Math.round(e.weight * 100) / 100;
	document.getElementById("length-strings").innerHTML = Math.round(e.length * 100) / 100;
	document.getElementById("cost-model").innerHTML = Math.round(e.cost * 100) / 100;
```

```
      // формуємо шлях для відображення моделі
      const reader = new FileReader();
      reader.onload = function() {
      init(reader.result);
      animate();
}
      reader.readAsDataURL(file);
    }
   })
  })
} else {
      document.getElementById("error").innerHTML = "Помилка! Можливо у вас застаріла версія
браузера!";
       }
}
```

Текст серверної частини

```
const express = require("express");
const path = require("path");
var multer = require("multer");
const cors = require("cors");
var bodyParser = require("body-parser");
var parseSTL = require('parse-stl');
var fs = require('fs');

var app = express();
app.use(cors());
app.use(bodyParser.json());
app.use(express.json());
app.use(express.static(__dirname));
app.use(express.urlencoded());

function calcVolum(pos) {
  var volum = 0;
  for (var i = 0; i < pos.length; i += 3) {
    var sum = (-pos[i + 2][0] * pos[i + 1][1] * pos[i][2] +
      pos[i + 1][0] * pos[i + 2][1] * pos[i][2] +
      pos[i + 2][0] * pos[i][1] * pos[i + 1][2] -
      pos[i][0] * pos[i + 2][1] * pos[i + 1][2] -
      pos[i + 1][0] * pos[i][1] * pos[i + 2][2] +
      pos[i][0] * pos[i + 1][1] * pos[i + 2][2]) / 6;
```

```
    volum += sum;
  }
  volum = volum / 1000;
  return volum;
}

var Storage = multer.diskStorage({
  destination: function (req, file, callback) {
    callback(null, "./models");
  },
  filename: function (req, file, callback) {
    callback(null, file.originalname);
  }
});

var upload = multer({ storage: Storage }).array("file", 3);

app.get("/", (req, res) => {
  res.sendFile(path.join(__dirname, "index.html"));
});

app.post("/calculate", (req, res) => {
  upload(req, res, function (err) {
    const fileName = req.files[0].originalname;
    if (err) {
      console.log("Ошибка: " + err);
      return res.send({ error: "Something went wrong!" });
    }
    const file = __dirname + '/models/' + fileName;
    var buf = fs.readFileSync(file);
    var mesh = parseSTL(buf);
    var pos = mesh.positions;

    if (pos.length % 3 != 0) {
       return res.send({
             success: false,
             message: "Error!"
       });
    };

    var model_volum = calcVolum(pos);
```

```
    var density = 1.04;
    var model_weight = model_volum * density;
    var length_strings = model_weight / 3;
    var unit_price = 2.25;
    var cost_model = length_strings * unit_price;

    return res.send({
      success: true,
      name: fileName,
      volum: model_volum,
      weight: model_weight,
      length: length_strings,
      cost: cost_model
    });
  });
});

app.listen(3000, () => console.log("Running at Port 3000"));
```

Текст створення сцени, камери та візуалізатора

```
var model, camera, scene, renderer, controls;
var w = 500;
var h = 500;

function init(model_name) {
    // scene
    scene = new THREE.Scene();

    // camera
    camera = new THREE.PerspectiveCamera( 75, w / h, 0.1, 1000 );
    camera.position.z = 100;

    // controls
    controls = new THREE.TrackballControls(camera);
    controls.addEventListener('change', render);
    scene.add( camera );

    // light
    var directionalLight = new THREE.DirectionalLight( 0xffffff );
    directionalLight.position.x = 0;
    directionalLight.position.y = 0;
    directionalLight.position.z = 1;
    scene.add( directionalLight );
    // add material
    var material = new THREE.MeshLambertMaterial({
```

```
        color: 0x00ff00,
    });

    // object
    var loader = new THREE.STLLoader();
    loader.load(model_name, function ( geometry ) {
        model = new THREE.Mesh( geometry, material );
        // add model at scene
        scene.add( model );
    } );

    // renderer
    renderer = new THREE.WebGLRenderer();
    renderer.setClearColor(0xffffff);
    renderer.setSize( w, h );
    document.body.appendChild(renderer.domElement);

    animate();
}

function animate() {
    requestAnimationFrame( animate );
    controls.update();
}

function render() {
    renderer.render( scene, camera );
}
```

Текст файлу package.json з вказаними залежностями

```
{
  "name": "express-html",
  "version": "0.0.1",
  "scripts": {
    "start": "nodemon server.js",
    "prod": "node server.js"
  },
  "dependencies": {
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "express": "^4.11.0",
    "multer": "^1.4.1",
    "parse-stl": "^1.0.2"
  },
  "devDependencies": {
    "nodemon": "^1.18.6"
  }
}
```

ДОДАТОК В

Хмарний сервіс швидкого прототипування

Опис програмного коду

УКР.НТУУ”КПІ імені Ігоря Сікорського”_ТЕФ_АПЕПС_ТР5276_19Б

Аркушів 8

Київ 2019

АНОТАЦІЯ

Даний додаток містить опис хмарного сервісу швидкого прототипування. Створений програмний продукт розраховує витрати матеріалу при друці, основні характеристики 3D моделі та виконує такі завдання:

– розрахунок об’єму звантаженої моделі;

– розрахунок маси надрукованої фігури ABS пластиком;

– розрахунок довжини нитки для друку моделі;

– розрахунок приблизної вартості 3D друку даної моделі;

– отримання завантаженої моделі у окремому вікні для перегляду;

– можливість взаємодіяти з відображеною моделлю (змінювати відстань до камери, огляд фігури з різних сторін) використовуючи комп’ютерну мишу.

Даний сервіс було написано на платформі Node.js, використовуючи технологію Ajax, мову розмітки HTML, мову стилів CSS, мову програмування JavaScript та бібліотеку Three.js.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис хмарного сервісу швидкого прототипування. У додатку Б міститься програмний код головних частин розробленої системи.

Для роботи з розробленим додатком необхідно мати доступ до мережі Інтернет, веб-браузер з ввімкненим JavaScript та модель у форматі STL.

При розробці програмного продукту використовувалась платформа Node.js, мова програмування JavaScript та середовище розробки Sublime Text 3.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблені компоненти виконують завдання розрахунку об’єму завантаженої користувачем 3D моделі, її маси, довжини нитки необхідної для друку та приблизної вартості друку ABS пластиком. Це відбувається за допомогою представлення трикутників, з яких складається модель, у вигляді тетраедрів.

Розроблений додаток може використовуватися на різноманітних підприємствах та виробництвах в якості обчислювальної програми.

Функціональні обмеження на використання додатку полягають лише форматі завантаженої моделі та її доцільності.

3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Для забезпечення повноцінної роботи та досягнення високої точності роботи хмарного сервісу швидкого прототипування було обрано платформу Node.js, використання якої дозволяє максимально просто втілити архітектурний стиль REST, а також має досить швидку обробку вхідних запитів користувача.

Також для роботи з 3D графікою у браузері використовувалися додаткові JavaScript бібліотеки Three.js, STLLoader.js та TrackballControls.js.

Розроблений додаток працює у браузері з увімкненим JavaScript та доступом до мережі Інтернет. Для роботи з ним користувачу необхідно мати 3D модель у форматі STL розрахунок вартості якої необхідно порахувати.

4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для реалізації хмарного сервісу швидкого прототипування було розроблено спеціальний алгоритм. Він будується на основі Ajax-запитів, які дозволяють обмінюватись клієнту та серверу даними без перезавантаження сторінки та з максимальною швидкістю.

Загалом, всі обчислення базуються на розрахунку об’єму моделі, яке полягає у її представлені у вигляді трикутників для подальшого обчислення їх об’єму у вигляді тетраедрів.

Під час запуску сервісу перед користувачем з’являється головне вікно для вибору моделі. Після того, проводиться перевірка формату завантаженого файлу та наявність артефактів. Якщо файл є недоцільним для розрахунку, то користувачу виводиться повідомлення про помилку в протилежному ж випадку з’являється вікно з моделлю та проведені розрахунки.

5. ВИКЛИК І ЗАВАНТАЖЕНЯ

Розроблена система не потребує додаткової інсталяції. Для того, щоб з нею працювати потрібно лише доступ до мережі Інтернет, веб-браузер з увімкненим JavaScript та модель для розрахунку витрат матеріалу у форматі STL.

Після запуску користувач переходить до головної сторінки сервісу де і вибирає необхідний файл з моделлю.

6. ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для розробленого додатку є файл з 3D моделлю у форматі STL.

Вихідними даними є розрахунок об’єму моделі, її маси, довжини нитки для друку ABS пластиком, приблизна вартість та вікно з завантаженою моделлю з можливістю змінювати відстань до неї та її розташування використовуючи комп’ютерну мишу.